Technical Report
CMU/SEI-88-TR-35
ESD-TR-88-036
November 1988

Experiment Planning for Software Development: Redevelopment Experiment

J. M. Perry
K. C. Kang
S. Cohen
R. Holibaugh
A. S. Peterson

Application of Reusable Software Components Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Title :Experiment Planning for software Development: Redevelopment Experiment

POC:Pat Mawby

Date:3 Feb 1998

This technical report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Karl H. Shingler SIGNATURE ON FILE
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Reviewed and edited by Information Management, a function of the Technology Transition Program, Software Engineering Institute.

Experiment Planning for Software Development: Redevelopment Experiment

Abstract: The Application of Reusable Software Components Project (ARSC) formulated an experiment design, data collection plan, and procedures in preparation for a reuse experiment. The reuse experiment is currently in progress and the experiment planning and the results to date are presented. While the design, plan, and procedures were developed to support the investigation of software reuse, they, as well as the process by which they were formulated, are applicable to any software development effort. They can be adapted to other technology investigations or to project-specific goals for improvement.

1. Introduction

Data collection on a software development project can be a time-consuming and costly effort. If conducted without proper planning it may result in a large volume of data of questionable value. Proper experiment planning identifies the data that will be sufficient to achieve explicit goals, provides efficient data gathering methods, estimates the data collection overhead, and provides methods for the validation and analysis of the data. To these ends, an experiment-planning framework is presented as a by-product of the reuse-based software development.

In the sense that a software development can provide empirical information for the improvement of development capabilities, every software development should be regarded as an "experiment." Other benefits of this view are that the experience and lessons learned from the development can be transferred to other projects, and that answers to project-specific questions can be obtained. Every software development plan should, therefore, address a project experiment plan. Specific goals of the experiment should be established and data collection planning for those goals should be performed. The set of data identified for consideration for collection, the set of data actually selected for collection, along with the rationale for selection, reflect the degree of project planning for improvement of software development capabilities.

The Application of Reusable Software Components Project (ARSC) incorporated an "experiment" into the redevelopment effort to investigate the impacts of software reuse on the software development process and products. This document reports on the experiment design and data collection effort of this project. An overview of the project and the reuse redevelopment experiment are included in Chapter 2. The experiment planning and the data collection mechanism and validation procedures are discussed in Chapter 3 and Chapter 4, respectively. Chapter 5 summarizes the results to date. This report concludes that experiment planning and data collection are important mechanisms for process improvement and technology transition and that they should be a standard part of every software development effort.

2. Project Background

Much of the literature and current work in software reuse focuses on techniques or tools for software reuse. These include components libraries, program generation, standards, design tools, and programming language features. Less work is available that addresses the application of these approaches on an actual development. Moreover, many of these proposals confine themselves to part of the development process or to particular software artifacts. There is a need for work on a more comprehensive or integrated approach to reuse and a need for empirical data on the application of software reuse; these needs motivated the Software Engineering Institute (SEI) to create the Application of Reusable Software Components Project.

2.1. Project Objectives

The reuse-based redevelopment effort is a major task of the ARSC Project. This project has the following objectives:

- Gain practical experience with state-of-the-art reusable software components, methods, and tools, and capture lessons learned in their application.
- Assess the impact of software reuse on software development activities and product.
- Identify and validate information that facilitates software reuse during system development.

The project will accomplish these objectives by the construction of a reuse testbed and by the reuse-based redevelopment of an mission-critical computer resource (MCCR) application in an experimental setting.

The project involves the redevelopment of major subsystems of a missile guidance system using reusable software parts and tools. The rationale behind the choices of application domain and the particular software reuse techniques is described in [10]. In brief, the missile guidance domain was chosen because it is a real MCCR application for which program office support is available, for which reusable parts and reuse tools have been developed, for which domain expertise can be obtained, and for which a Program Performance Specification (PPS) document [8] is available. The software parts and tools selected for this experiment are the Common Ada Missile Packages (CAMP) [4]; EVB [6] and Booch [3] parts; CAMP/AMPEE (Automated Missile Parts Engineering Expert) System [1]; and GTE ALS (Asset Library System) [7] [9].

2.2. Reuse-Based Redevelopment

The reuse-based redevelopment [11] is a single team, single project experiment. It is an empirical study of software reuse, collecting goal-directed data during the development of a real-time system. Starting with the PPS, the project will develop the navigation, guidance, and autopilot subsystems utilizing CAMP, EVB, and Booch parts using the CAMP/AMPEE and GTE ALS tools.

The CAMP parts originated with the analysis of ten systems and their requirements. This analysis identified requirement, subsystem, and module commonality, which became the basis for the CAMP parts.

The reuse project is redeveloping the guidance, navigation, and autopilot subsystems of one of the ten original systems that were included in the CAMP commonality analysis. The experiment will utilize only the PPS from one of the original systems; hardware and the programming language are different from the original system, and the design will be new, incorporating reuse. A redevelopment of one of the original systems has the advantages of providing a high degree of reuse for the development, having available customer experience for the evaluation of the results of the experiment, directly addressing the use of the specialization of generalized parts (thus, providing the natural setting for evaluating the use of the parts, see Figure 1 below), and enabling the possibility of comparisons with the original system.

The redevelopment of one of the original systems leads to the concept of the reuse cycle shown in Figure 1. The top path of the figure corresponds to development without software reuse, with the rest corresponding to development with reuse. The branch into the top path, labeled-use, corresponds to new development that uses the reusable parts and generic subsystems. The branch to the right (also labeled use) corresponds to redevelopment of existing systems using the reusable parts and generic subsystems. Our experience on the reuse experiment suggests that reuse-based redevelopment as shown in this figure may be a necessary step in the development and evolution of reusable parts, like CAMP, and a helpful step that can be exploited to initiate the formation of a generic model of the application.

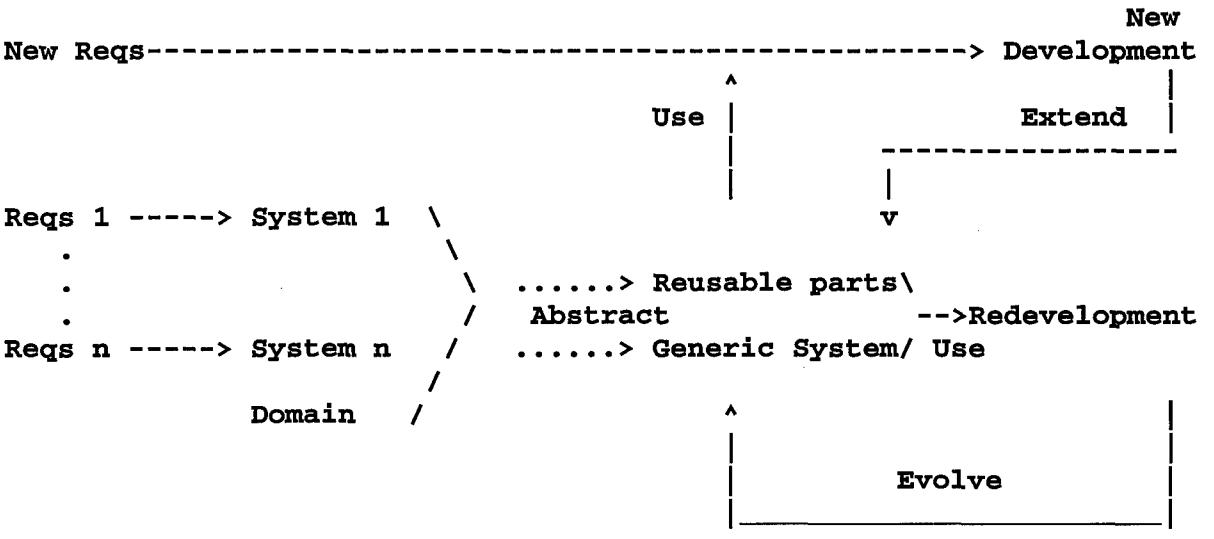

Figure 1. The Reuse Cycle

2.3. Software Reuse Assumptions

The application of reuse on a software development project is influenced by assumptions on the nature of reuse itself and on the approach used in its practice. It is helpful to make these assumptions explicit since they influence the goals of the experiment, the experiment planning, and data collection activities.

Software reuse is the practice of systematically acquiring software solutions, representing them, and applying them to new development problems. One approach for applying reuse consists of the steps of specifying problems; searching, assessing, and comparing candidate solutions; selecting, modifying, and integrating solutions to the problems; and evaluating the final results. These steps follow from the view of software reuse as a problem-solving strategy. Reuse-based development is essentially analogical problem solving, i.e., analogical development, where a new development is based on a previous development having similar requirements. Given a problem of the form "develop a system," problem solving begins with the selection of a solution model that includes an architectural framework. It is assumed that the selected framework has associated reuse forms, such as a library of system packages, of reusable application parts, and of subsystems. The solution system is derived from the model by successive refinement, through construction of new parts, and by the integration of reusable ones. Each refinement begins with a specification of subproblems. This specification is used as is, or modified as necessary, and used to search the reuse libraries. The search results in a collection of reusable parts, which are assessed for

application to the derivation of the subsystem. If appropriate, reusable parts are selected and modified, when necessary. The reusable parts are then integrated into the model. The resulting refinement is evaluated with respect to implementing the specification and adhering to any imposed constraints.

Our notion of software reuse includes three approaches:

1. The use of application-independent software systems, for example, a database system and database commands to satisfy requirements for a data storage and retrieval system.
2. Derived development where features of a base product are removed, added, or modified to satisfy new requirements.
3. Construction whereby the software system is built out of collections of layers, subsystems, and application-independent parts.

In practice, instances of each of these approaches are utilized, with the third approach being the most common. To promote them, a reuse-based software development methodology [12] was developed and tailored [13] to integrate and adapt them to the reuse redevelopment effort.

The tailored reuse methodology reflects our assumptions on the nature of reuse. These assumptions will be evaluated using the data collected during the experiment.

3. Experiment Planning

3.1. Planning Frameworks

The redevelopment experiment is being done within the goal/question/metric framework developed and promoted by NASA's SEL [2]. In this framework, the actual application development is preceded by experiment definition (goal definition) and planning. The framework consists of six steps:

1. Establish goals of the experiment.
2. Identify questions of interest that derive from the goals.
3. Identify appropriate data categories.
4. Design and test data collection instruments.
5. Collect the data and validate it.
6. Analyze and interpret the data.

The goal/question/metric framework is summarized in Figure 2.

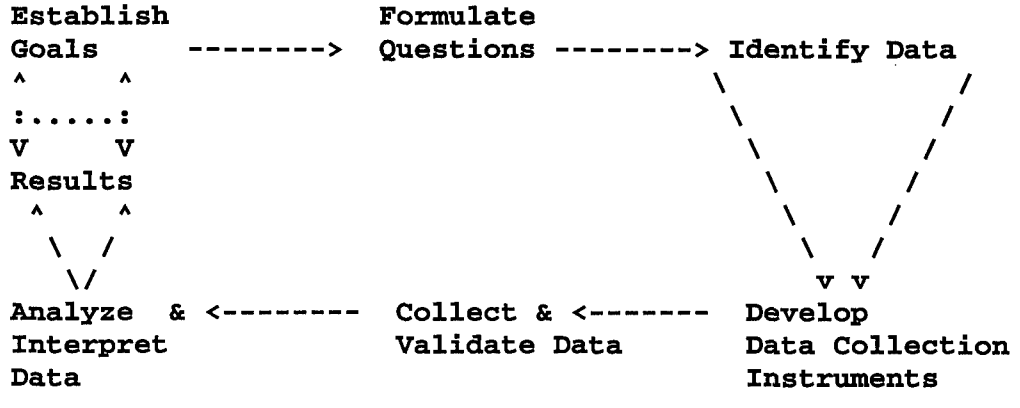

Figure 2. Goal/Question/Metric Framework

The results of a software development effort are affected by factors that pertain to the nature of the software development process (tasks) and products (software artifacts). A factor of particular interest in this investigation is the reuse-based software development. The reuse-based software development is influenced by the approach that is adopted for practicing software reuse. The reuse approach, in turn, follows from certain beliefs, expectations, and assumptions regarding software reuse; these are expressed as a thematic hypothesis.

Software development factors (artifacts and tasks) and a theme were incorporated into the goal/question/metric framework to create a reuse experiment framework. Based on the theme, experiment goals were formulated and relevant factors were identified. These were used to formulate questions that, in turn, were used to identify data needed to answer the questions and accomplish the goals.

The reuse experiment framework is depicted in Figure 3.

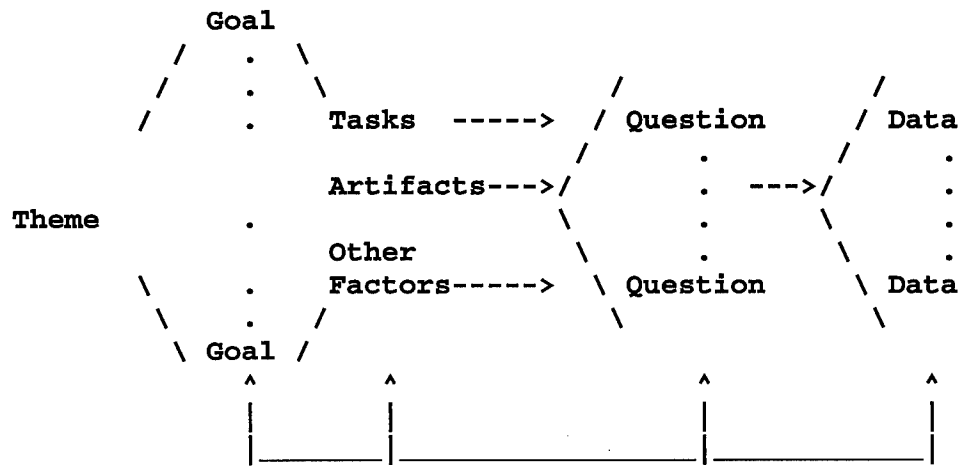

Figure 3. Reuse Experiment Framework

Following the experiment framework, the reuse-based redevelopment project was planned. The project-specific theme, goals, factors, and questions were defined, and the data to support the questions were identified. These are presented in the following sections.

3.2. Reuse-Based Redevelopment Experiment Planning

3.2.1. Theme

Preparatory work of the reuse project involved extensive literature and conference investigation on software reusability, and on software engineering experimentation and data collection. The scope of this preparatory investigation is embodied in a project bibliography [10].

From this preparatory investigation and based on the project team's experience, a view of the state-of-the-art and state-of-the-practice of software reuse was arrived at. This background research identified a diversity of work pertaining to software reuse spanning a diversity of topics, both managerial and technical, including reuse libraries, economics, tools, techniques, and methods for achieving reuse. Our context for considering this diversity of

work was that of a software development project; that is, if our goal was building a software system exploiting reuse, how should we proceed? What resources would we need? What problems relating to reuse would be encountered? We categorized work from the literature in terms of reuse resources (i.e., reusable components, tools and methods), the development of reuse resources, and the application of these resources. Given our context, we were particularly interested in the latter, but work and data on applications of reuse were rare. Our first step was to identify critical elements that would affect the application of reuse. These were:

- Availability of reusable resource: artifacts from requirements through code; tool support for working with the artifacts.
- Attributes of reuse resources: domain dependence/independence, level of abstraction, applicability to the application, maturity, and formality.
- Maturity of the body of knowledge for solving problems and building systems in the application domain(s) of interest.
- Existence of a model, framework, methodology, paradigm, or integration concept for guiding the application of reuse throughout the software development.
- Learning necessary to utilize the reuse resources for the domain(s) of interest.

This view of software reuse was expressed as a hypothesis that we called the theme of the experiment. The nature of the experiment, i.e., single team/single development, is such that it will produce data that can help others prepare for potential problems in their application of software reuse. The theme is stated below:

Theme. Current software reuse technology can have a significant positive impact on software development and be more effectively utilized if:

1. There is a sufficiently rich and powerful collection of reusable components, reuse methods, and tools for the application.
2. The domain of application is sufficiently mature and well understood so that there is a standard model for a class of systems in that domain.
3. There is a basis—in terms of a model, paradigm, or concept—that supports a reuse methodology for integrating and systematically applying various reuse methods.

The effort to effectively utilize reuse will initially be significant. The learning curve of project personnel will increase due to reusable components, reuse methods, and tools; and there will be major adjustments to the requirements, design, and integration phases. These changes will introduce new problems in the development. Moreover, improvements in cost and productivity from software reuse will not necessarily occur for a single project, but will result when cost and effort are amortized over several system life cycles.

3.2.2. Goals

The goals of the reuse redevelopment experiment were established based on the project objectives and the thematic hypothesis discussed earlier. They are to:

1. Formulate and improve a reuse-based methodology for software development.
2. Gain experience and lessons learned in the application of the reuse-based methodology of the CAMP, EVB, and Booch parts, and of the CAMP and GTE tools.
3. Assess the impact of software reuse on software development activities and products, particularly design.

The methodology defines reuse-based software development life-cycle activities, identifying where in the lifecycle reuse activities occur. Lessons learned include problems and difficulties encountered in the application of the methodology and reuse forms, and any solutions or recommendations to resolve them. Impact refers to changes to development activities and their associated costs, effort, benefits, or shortcomings.

Based on the experiment goals, any process, product, and environment factors that affect the outcome of the experiment were identified. These factors are described in the next section.

3.2.3. Factors

Literature on experiments and data collection for software development, as well as our own understanding of the development process, identified factors that affect the software development. These factors are:

- Characteristics of the development team
- The development environment
- The software engineering technology, including reuse technology and the methodology for applying it during software development
- Customer involvement in the development
- The application domain and system characteristics

To address the goals of the study, we examined the software process factors relative to the theme and the goals. This resulted in the need to identify reuse-related tasks and artifacts, which in turn raised the need for a methodology for applying reuse. The task and artifact factors are discussed below; the other process factors are discussed in the high-level and detailed plans for the reuse experiment [14], [10].

The methodology for applying reuse is an extension of [5] with refinement of each phase to identify reuse activities. The reuse activities that are common across the life-cycle phases are identified as:

- Studying the problem and available solutions to the problem, and developing a reuse plan or strategy.
- Identifying a solution structure for the problem following the reuse plan.

- Reconfiguring the solution structure to improve reuse at the next phase.
- Acquiring, instantiating, and/or modifying existing reusable components.
- Integrating the reused and any newly developed components into the products for the phase.
- Evaluating the products.
- Identifying the components that are reusable.

These activities are used as the base model for defining the specific activities at each phase of the life cycle. Artifacts of the methodology are those typically found in a traditional phased software development, like [5]. The reuse methodology is general; tailoring of the methodology for the project is described in [13].

Specific questions that are relevant to the experiment goals and that are directly addressed in the experiment were formulated considering the experiment factors. They are described in the following section.

3.2.4. Experiment Questions

A large amount of effort was devoted to the formulation of objectives in the form of questions. The process for producing the questions involved joint work by the project team over many meetings, and individual analysis by team members on assigned questions. A candidate list of twenty questions was generated from an examination of the implications of reuse on software development activities and from consideration of the factors from the perspective of software reuse. The candidate questions were subjected to examination and analysis.

The candidate questions were categorized into four groups: Usage, Product, Process, and Application Domain. The categories were derived from the goals and theme of the experiment. Then the questions were mapped to the goals. The classification and mapping of the questions provided information on the coverage of the questions with respect to the goals and on the relationships among the questions.

These twenty questions were then applied to three candidate application systems, each involving specific reuse resources. The three were: navigation and guidance subsystem/CAMP, LAN control subsystem/TCP-IP generics, and inertial navigation subsystem/data structure parts. These three subsystem/reuse resource combinations were those that were available to us at the time. For each of these domains and reuse resource combinations and for each question, the data needed to answer the question were identified and a possible answer was conjectured. Attempting to answer each question in the context of a specific domain and specific reusable parts and reuse methods/tools provided information on the answerability of each question and suggested data collection forms needed for gathering the data to answer that question.

Questions were eliminated on the basis of interest, answerability, or importance and relevance to the theme and goals. Similar questions were combined or used to generate a single question. The resulting list consisted of nine questions. The nine questions were allocated to the five team members, one or two questions to an individual, for question analysis. The analysis of a question included research,

draft, team meeting for team consensus, and final report. The reports included definitions of relevant terms, discussion of the question, identification of data needed to answer the question, metrics, data collection forms, and the type of analysis to be performed on the data. The analysis reports were reviewed and consensus was reached on the definitions, scope, and data for each question.

The nine questions that resulted from the question formulation process are summarized below. Where necessary, annotations are given.

1. What is the extent and frequency of use of the available reuse forms (reusable parts, reuse methods, and reuse tools)?

   The scope of this question includes both process and product, and extent and frequency can be interpreted with respect to both of these dimensions. For a product, "use" pertains to both derivation and operation. Derivation of a product is the construction of the product from reusable parts. Operation is the execution of the product in an operational environment. "Extent" is a measure of how widespread the use is, i.e., where; and "frequency" is a measure of how often the use occurs, i.e., when.

2. How much training, experience, and external support for the subsystem application domain or reuse was needed to use the available reuse forms?

   The purpose of this question is to understand how dependent the software development is on domain knowledge and reuse techniques. In the context of our experiment, this question becomes: how much training, experience, and external support was needed to apply the reusable components (CAMP, EVB, Booch), the CAMP/AMPEE constructors, and the library tool (ALS) in the development of a missile guidance system?

3. What is the relative contribution and value of the available reuse forms; and what was the effort to use them relative to the total system development and final products?

   Value is a quality indicative of the degree to which a part, method, or tool is useful with respect to the software system and its development. "Value" can be relative to formulating a problem, solving a problem, or achieving desired properties of the system.

   This question seeks to identify the more useful parts, methods, and tools, as well as those that are least useful, and, ultimately, the reasons, why. Of special interest is the identification of groups of parts that are utilized together and the ways in which they are interconnected to comprise a single entity. Question 3 is the qualitative counterpart to Question 1.

4. What requirements and design decisions were related to the reusable forms?

   The above question addresses the role the reusable parts, reuse methods, and tools play in design decisions. A design decision involves requirements, constraints, choices, frames of reference, and artifacts. The scope of the question includes which reuse forms were related to these design decision elements and what the nature of the relationship was.

5. What are the attributes, static and dynamic, of the resulting system and its design?

6. What is the impact of using the reuse forms on the system quality factors (adaptability, performance, reliability)?
7. Is there a standard, application model, paradigm, or integral concept that forms the basis for a reuse methodology for the missile subsystem domain? If so, what are its limitations or strengths for promoting reuse?

   Our assumptions concerning approaches to reuse suggest the formulation of a generic architectural application model and its use as a context for the derivation of the system under development. The tailored reuse-based methodology employs such a model. This question addresses the degree of evolution and refinement of the model and its effectiveness in promoting reuse.
8. How effective were the Common Ada Missile Packages (CAMP) components and tools in rebuilding one of the source systems (the source systems are the ten missile subsystems used in the CAMP commonality study)? How does the reuse-based developed subsystem compare with the original system?

   The interest of this question is in the generalization/specialization process of the reuse cycle depicted in Figure 1 above and the collection of parts that were used to construct the system, as well as the parts that were not used in the construction.
9. What are the distributions of "standard" data (effort, cost, time, changes, errors) with respect to various dimensions (tasks, product/structure, phases, reuse/non-reuse, application dependent/independent, time)?

Question formulation and the analysis process identified the data required to answer the questions. The identified data were classified and organized into forms. The data collection forms and collection procedures are discussed in the following chapter.

4. Data Collection and Validation

4.1. Forms

A candidate list of data that supports the questions was generated. From the candidate list, the final list was generated considering the relevance to the experiment goals, the resources required to collect the data, redundancy, collection difficulty, and measurability. The selected data were categorized according to the time of collection and process, product, and development-team orientations. The data categories are:

1. reuse
2. team background
3. design decision
4. external support
5. product attributes
6. change/error
7. cost
8. task

These data are collected using seven forms: Daily Activity, Design Decision, Reuse Process, Support, Training, Error/Change, and Personnel Background. There is currently no form to collect data on product attributes. This data will be collected during post mortem. Cost data will be derived from activity data. The forms are summarized below. Copies of the forms appear in [10].

The Daily Activity Form collects effort data by time spent on non-project, related activity and project-related activity. Project-related activity is classified into the categories of experiment-related, training, management, domain, reuse, and task (of the methodology). This form contains pointers to other forms to collect detailed data on specific activities, e.g., training.

The Design Decision Form collects data on high-level design decisions and on major detailed design decisions. Data collected include the problem or decision being addressed, the resources used, the resources that are not used, constraints that apply, constraints that do not apply; partitioning, data structure, and algorithm alternatives; choices made and corresponding rationale.

The Reuse Process Data Form collects reuse usage data on the reusable parts, reuse methods and tools. Categories of usage include specifying, searching, assessing, modifying, integrating, and evaluating. Attributes for usage include date and time, duration, frequency, objects, mechanism, context, and results.

The Support Data Form focuses on dependency on domain knowledge and reuse techniques provided by non-project team members from outside organizations. Data identify the parties involved, the nature and duration of the support, and the results.

The Training Form captures data on pertinent training to supports the development. This includes domain and reuse-related training. The data consist of who provide the training, who attends what the training is, how long it lasts, where it is provided, the cost, schedule, and relevance.

Error and Change Forms are typical configuration-control forms that have been adapted to the reuse redevelopment. They have been extended to cover modifications made to accommodate reuse and to cover errors related to the reusable parts, reuse methods and tools.

The Personnel Background Form records, in particular, the domain, reuse, Ada, and software engineering experiences of the team members.

The forms are interrelated by pointers that provide traceability among the forms. For example, traceability enables the gathering of the design decisions, external support, reuse resources, and errors/changes pertaining to a specific requirement.

4.2. Procedures

Data are collected using paper forms that are controlled by a data collection administrator. The entries called for on the forms follow from an elaboration of each of the nine questions above, with respect to the meaning of the terms employed in the question and the defined scope of the question. The forms are coded, distributed, collected, and entries are validated by the administrator. The Daily Activity Form is filled out by the project team members on a work-day basis; the other forms are filled out on an activity basis. The collected forms are compiled into monthly notebooks and filed. Data from the forms will be entered, at some future time, into an Ingres database by the administrator. The Ingres database will support the analysis to answer the experiment questions.

Data are currently kept in notebooks of completed data collection forms. They are collected and grouped by month by type of form. The total time spent by each team member per entry per data form is accumulated as the completed form is validated. This time summary is kept in the notebooks. The notebooks are filed in the data-base administration office.

Plans call for the data to be entered into an Ingres database. An Ingres form has been defined for each manual data collection form. The forms will be accessed from a main menu that will allow updating and reporting. Once a form has been filled in and completed, it will be closed; otherwise it remains open. Open and closed forms can be reported on by user, date, or form type.

The mapping of the data to a database design requires careful consideration with respect to the definition of the records and fields. Names, display format, display attributes, type of field, default value, and relationships between the data collection forms have been identified.

5. Experience and Issues to Date

5.1. Data Analysis

The data collected from February 1988 through August 1988 consists of daily activity data, training data, and personnel background data. Personnel background data was collected on each member of the project team. Training data was collected on STATEMATE, GTE ALS, CAMP, and VMS training. Daily activity data is summarized in Figure 4.

The data in Figure 4 is organized by month, from February (the start of data collection) through August. A pair of data is presented for each category. The data on the left is the percentage of time over the total work hours, and the data on the right is the percentage of time over the total work hours minus SEI overhead. The SEI overhead includes efforts that are considered unique for the SEI, for example, technology transition effort and the effort by the affiliates for their organizations. The project team consisted of six members for February through May, five members for June, and four members for July and August. Two members who left the project are industry affiliates.

Percent of effort-hours by category per month by a six member team

	February	March	April	May
Training	3/4	14/22	4/5	0/0
Communication	17/25	3/5	6/8	3/5
Experiment	22/31	7/12	10/13	4/6
Management	7/9	5/7	5/7	5/7
Data	2/3	0.6/1	0.6/1	1/2
Domain Study	0/0	0.3/0.5	2/3	3/5
Reuse Related	0/0	7/10	8/11	22/34
Task Related	14/20	20/31	28/39	18/28
Proj Overhead	6/8	8/12	9/12	9/13
SEI Overhead	30/-	35/-	28/-	35/-
Work hours	786	1057	982	950
Workdays	17	23	21	21

	June	July	August
Training	7/13	0/0	0/0
Communication	3/6	3/4	0/0
Experiment	2/3	1/1	0/0
Management	5/10	6/7	4/4
Data	6/12	2/3	2/2
Domain Study	1/2	2/2	0/0
Reuse Related	6/12	11/13	0/0
Task Related	14/28	22/27	55/62
Proj Overhead	7/14	36/43	27/31
SEI Overhead	49/-	17/-	12/-
Work hours	868	600	711

Workdays	22	20	23

Figure 4. Summary Data, 2/88 through 8/88

Training hours consisted of two three-day CAMP workshops, one two-day STATEMATE workshop, one three-day ALS workshop, a one-day VMS Workstation class, and a five-day Ada training attended by a project member. Most of the trainings took place in the first three months.

Communication is all forms of interaction, such as project meetings and conversations dealing with the development. The 17% expended here for February indicates a relatively high startup need.

Management is project management. The percentage of effort here remains somewhat constant over the months at about 5% level.

Experiment denotes experiment design and review. The experiment design was essentially completed prior to February. This category includes time spent for the review of the methodology task list and for refinement of the data collection forms and procedures. The 22% on experiment for February, together with the 17% on communication, indicates a relatively high startup cost of the experiment. There were frequent meetings to review and discuss the data collection forms and procedures.

Data includes data collection, validation, and analysis efforts. The Daily Activity Form takes little time to complete. It is filled out daily by each team member in units of hours or .5 hours. Typically the form takes less than 15 minutes to fill out. This time to fill out the form amounts to about 1 to 2% for the team. Hours for data validation and analysis are somewhat sporadic, due to changeover in data collection personnel. The data validation and analysis activity in June increased the overall data collection effort to 6%. A consistent effort on data validation and analysis is of paramount importance.

Domain Study includes efforts expended in learning the application domain. This effort primarily consisted of reading literature on missile systems.

Reuse Related work consists of classification of reusable components, and informal learning of CAMP and ALS tools, and of the CAMP, EVB, and Booch parts. Features analysis which is part of the tailored reuse methodology is done during the software requirements analysis phase to which significant effort began to be devoted in July and August.

Task Related is the set tasks of the development other than reuse related ones. This category, combined with the reuse related effort, indicates the effort which is directly devoted to developing the system. This combined effort was increasing from February (14%) through May (40%), decreased in June (to 20%) because of the Affiliates Symposium, was relatively low in July (at 33%) because of vacations, and significantly increased (to 55%) in August.

Project Overhead includes all non-developmental project activities that are considered typical in industry. This includes personal time, vacations, and sick-days. The high percentage of project overhead in July and August is due to vacation activity.

SEI Overhead is all non-developmental activity that is unique for the SEI mission. This category includes activities such as technology transition, work by the affiliates for their organizations, course work, and seminars. In June it included technology transition work done in preparation for the Affiliates Symposium, and the

centages for the other months as well. This non-project effort, thus, poses a relatively large "loss" of effort to the project.

5.2. Lessons Learned

Although the redevelopment effort is still in the design phase, a number of experiment and/or development problems have surfaced. Lessons pertaining to the experiment design and data collection are discussed below.

At the beginning of the experiment, there was a concern among project members that the Daily Activity form, which is filled in the unit of half an hour, would be intrusive to the development effort and that the data would be biased. However, the experience with the form by the members is that it is not as intrusive as initially thought. Project members usually record their daily activities twice a day, before lunch break and at the end of the day, and then complete the forms weekly, before submitting them to the data administrator. The effort expended on data collection was less than 2% of the total effort.

The primary data collection problem is the difficulty of collecting subjective data (and the reliability of it). The Daily Activity, Personnel Background, and Training forms are well defined, that is, the set of data they are intended to capture and the data they actually capture are expected to coincide and when each form is to be filled out is clear. However, the Design Decision, Reuse Process, and Support forms collect subjective data, and the forms tend to be too dependent on the individual and too complex, which often cause misinterpretation of the forms and the loss of data. Reuse data is not well understood and, as a result, there is a tendency for the form to ask for a large amount of data. Some external support and reuse activity has been reported on the Daily Activity form, and these should be detailed on Support and Reuse Process forms. Moreover, although the project is in software requirements, experience tells us that design decisions can still be made during this phase. In fact, the reuse-based methodology entails a look-ahead to following phases, in order to increase reusability during the later phases. Thus, one would expect there to be more data related to this area. This is manageable, but tells us that data analysis must be ongoing in order to address problems and to identify lessons, as soon as possible.

After data collection was underway, it became apparent that the data collection forms should be supplemented by a project journal which would record influential project incidents and help capture lessons learned. Without such a journal, these incidents may, at best, only be named on a daily activity form along with a time duration allocation, and the significance or insight of the incident left to memory. A project journal has since been instituted and a weekly interview of each project member is being conducted to capture significant events.

The issues of loss of data and need for a project journal are in part due to assignment of data administrator duties to part time personnel. This position has had two changeovers in six months and is currently vacant. Duties are currently assigned to a project secretary who has other responsibilities. The availability of a data administrator is essential to data collection and analysis.

Capturing process data requires a detailed software development methodology. This methodology is important, not only for guiding the development, but as a check list for assuring that all desired process data is being collected. The Daily Activity Form was used to collect effort data by methodology task. On correlating the data against the tasks of the methodology, one finds a lack of data for some tasks. This could indicate that some of the tasks called for in the methodology are not being done, the data on the tasks is not being collected, the tasks were not identified in the methodology, or the tasks are too detailed to separate. In the first case, there is a need for methodology enforcement or a change to the methodology. In the second case, the newly established interviews or completion of the proper data collection form will collect the data. In the last two cases, the methodology task list needs to be adjusted.

These problems and difficulties and those yet to arise, as well as their resolution, will help identify lessons to support the improvement of the data collection framework, the application of the reuse resources, and the methodology.

6. Summary and Conclusion

The Application of Reusable Software Components Project is investigating the impact of software reuse on the software development process and products in an experiment framework. The experiment framework is developed based on the goal/question/metric data collection framework, with adaptations for the reuse investigation. The instantiation of this framework for our reuse-based redevelopment is shown in Figure 5 below. It is a summary of the thematic hypothesis, goals, factors, questions, and data given previously.

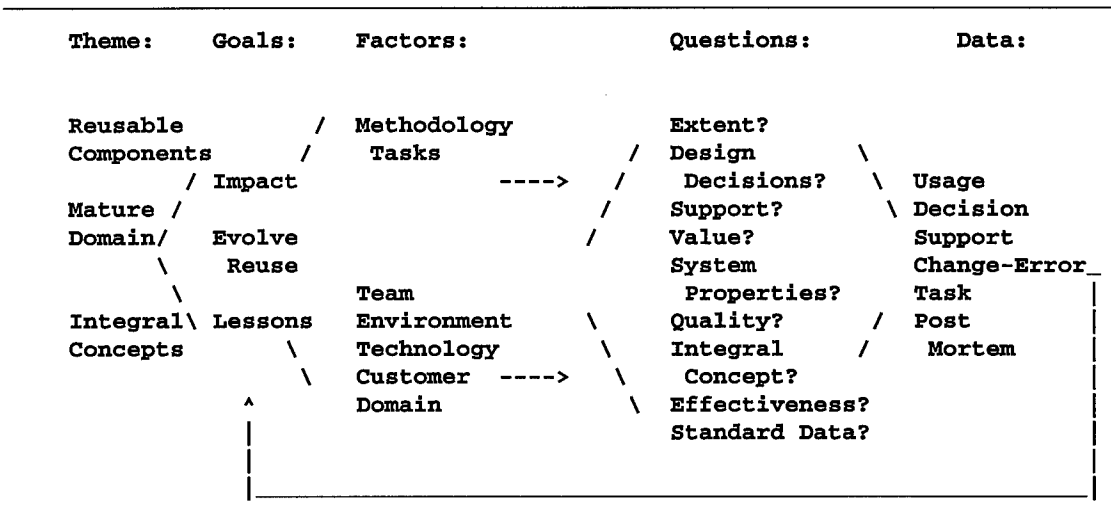

Figure 5. Reuse Experiment Framework Summary

The information collected from the experiment will help us to understand how the availability of reuse forms influences the development process and products, and to identify when reuse presents benefits and when not. The reuse experience from this experiment will place the SEI in a position to advise those who employ reuse-based development on how to structure the reuse-based development life cycle, which areas to emphasize or to avoid, and how to measure the effectiveness of their reuse initiatives. Organizations can use this data as a basis for estimating reuse costs and benefits. They can also use our experiment-planning framework as a means of planning their own data collection activity.

A software development effort can provide empirical information for the improvement of development capabilities. In this sense, every software development should be regarded as an "experiment," and experiment design and data collection planning should be a standard part of software development. A new technology, such as reuse, will mature through application, evaluation, and improvement.

References

[1] McDonnell Douglas Astronautics Co.
Software User's Manual for the Ada Missile Parts Engineering Expert Systems of the Common Ada Missile Packages (CAMP) Project.
McDonnell Douglas Astronautics Co., P.O.Box 516, At. Louis, MO 63166, 1987.

[2] Basili, V. and Selby, R.
Data Collection and Analysis in Software Research and Management.
Proc. American Statistical Association and Biometric Society Joint Statistical Meeting. :Pages 21-30, August, 1984.

[3] Booch, G.
Software Components with Ada.
Benjamin/Cummings, Menlo Park, CA, 1987.

[4] McDonnell Douglas Astronautics Co.
Common Ada Missile Packages (CAMP): Overview & Commonality Study Results.
McDonnell Douglas Astronautics Co., P.O.Box 516, At. Louis, MO 63166, 1985.

[5] DOD-STD-2167A.
Defense System Software Development (draft).
Military Standard, October, 1987.

[6] EVB Software Engineering, Inc.
Grace Notes.
EVB Software Engineering, Inc., 5303A Spectrum Dr., Frederick, MD 21701, 1986.

[7] Jones, G. and Prieto, R.
Asset Library System: Summary Report.
GTE Technical Note (No. 87-126.06):, December, 1987.

[8] McDonnell Douglas Astronautics Co.
Computer Program Performance Specification Cruise Missile Land Attack Guidance System BGM-109C.
McDonnell Douglas Astronautics Co., P.O.Box 516, At. Louis, MO 63166, 1985.

[9] Prieto-Diaz, R. and Freeman, P.
Classifying Software for Reusability.
IEEE Software, Vol. 4(No. 1):6-16, January, 1987.

[10] ARSC.
An Experiment to Analyze a Reuse-Based Software Development: Detailed Design.
Software Engineering Institute (technical report in preparation), December, 1988.

[11] Holibaugh, R., Perry, J. and Sun, A.
Subsystem Redevelopment: Analysis.
Software Engineering Institute (CMU/SEI-TR-88-014), November, 1988.

[12] Kang, K.C., Cohen, S., Holibaugh, R., Perry, J. and Peterson, A.S.
A Reuse-Based Software Development Methodology.
Software Engineering Institute (technical report in preparation), November, 1988.

[13] Perry, J.
Perspective on Software Reuse.
Software Engineering Institute (CMU/SEI-TR-88-022), November, 1988.

[14] ARSC.
An Experiment to Analyze a Reuse-Based Software Development: High-Level Design.
Software Engineering Institute (technical report in preparation), November, 1988.

Table of Contents